KEYBOARD FOR AN ELECTRONIC WRITEBOARD AND METHOD

Field Of The Invention

[0001] The present invention relates to electronic writeboards and in particular to an on-screen keyboard for an electronic writeboard.

Background Of The Invention

[0002] Electronic writeboards or whiteboards (EWBs) are known in the art and have been used in conjunction with host computers executing applications software to provide enhanced multimedia and teleconferencing capabilities. An example of an electronic writeboard of this nature is sold by SMART Technologies Inc. of Calgary, Alberta, Canada under the name SMART Board. The SMART Board electronic writeboard includes a touch sensitive panel, a tool tray accommodating a plurality of tools such as colored pens and an eraser as well as a driver and an associated controller.

[0003] In use, the electronic writeboard is connected to a host processor such as a personal computer operating in a Windows® environment and executing applications software, via a serial data connection. The electronic writeboard can be operated in one of three modes, namely a projected mouse mode, a projected mark-up mode and a non-projected mode.

[0004] In the projected mouse mode, the image displayed on the monitor of the personal computer is projected onto the touch sensitive panel. In this case, the electronic writeboard functions as a giant mouse providing input to the personal computer in response to user contact with the touch sensitive panel. Specifically, the electronic writeboard generates mouse events in response to user contact with the touch sensitive panel which are conveyed to the personal computer for processing. Thus, by contacting the touch sensitive panel, the personal computer can be conditioned to open and display menus, to activate displayed menus, to drag icons, to execute software and to switch applications by changing input focus. The latter event of course is achieved by contacting the touch sensitive panel outside of the window of the active application running on by the personal computer.

[0005] In the projected markup mode, the coloured pens and eraser are used to contact the touch sensitive panel. The active application running on the personal computer tracks where writing and erasing has occurred and maintains a computerized image of what is drawn on and erased from the touch sensitive panel. The computerized image is projected onto the touch sensitive panel so that user can see the computerized image.

[0006] In the non-projected mode, there is no image displayed on the whiteboard by the computer. Contact made on the whiteboard is recorded on the attached computer. Since the whiteboard is in non-projected mode, it is not interactive since users cannot see how their strokes are being recorded. For this reason, in non-projected mode, people write on the whiteboard with standard dry-eraser markers. The computer then tracks their writing and allows them to save the notes later.

[0007] To enhance user input abilities, on-screen keyboards for use with electronic writeboards have been considered. For example, Innovative Management Group Inc. of California, U.S.A. has developed an on-screen keyboard sold under the name My-T-Touch. The My-T-Touch keyboard is an extension of a touchscreen interface and uses “Heads Up Display” technology designed to keep a user’s focus and concentration in one place. Thus, visual re-focusing and re-positioning, caused by the up and down motion of going from screen to keyboard to screen is reduced.

[0008] Unfortunately, prior art on-screen keyboards do not address certain technical issues and therefore, suffer a number of problems. For example, some computer platforms such as Microsoft Windows® always ensure that an active application has input focus. In other words, these computer platforms ensure that any application that is selected through a mouse click receives input focus, i.e. the application becomes active. Thus, when a user contacts a prior art on-screen keyboard displayed on a touch sensitive panel to enter text into an active application, input focus is switched from the active application to the on-screen keyboard.

[0009] Unfortunately, only the application with input focus may receive inputs from peripheral devices, such as a mouse or a keyboard. Therefore input focus must be removed from the on-screen keyboard and given back to the application in order for the application to receive any input from the on-screen keyboard. Switching input focus from the on-screen keyboard to the application each time the on-screen keyboard is touched causes many unpleasant side effects. Firstly, a change in the activation state of an application causes the application to refresh or redraw portions of itself. This means that every time input focus is changed from the active application to the on-screen keyboard and then back to the application, the application will flicker as it refreshes. Secondly, a loss of input focus causes the application to close all of its open menus. As a result, prior art on-screen keyboards cannot be used to navigate through the menus of an active application. Finally, some applications, such as Internet Explorer^{TM}, do not retain the text insertion point whenever input focus is lost. This means that when input focus is switched from the active application to the on-screen keyboard and then back to the application, the cursor is often positioned at a different point than where the cursor was prior to the active application losing input focus. In some circumstances, this makes it impossible to type text into certain fields using an on-screen keyboard.

[0010] It is therefore an object of the present invention to provide a novel on-screen keyboard for an electronic writeboard and an interactive display system incorporating the same.

Summary Of The Invention

[0011] According to one aspect of the present invention there is provided an electronic writeboard for communicating with a computer including applications software and running an active application comprising:

a touch sensitive panel on which the screen image output of said computer is displayed, said touch sensitive panel being responsive to user contact and generating events;

a keyboard window displayed on said touch sensitive panel and including a keyboard having a plurality of user selectable keys;

a driver receiving said events, said driver sensing user contact on said touch sensitive panel within said keyboard window and generating messages in response thereto; and

a controller executing a keyboard application and receiving said messages, said keyboard application processing said messages to provide data to the active application running on said computer corresponding to keys of said keyboard contacted by said user.

[0012] In a preferred embodiment, the driver passes events resulting from user contact on the touch sensitive panel outside of the keyboard window directly to the computer for processing. During initialization, the controller registers with the driver and provides a keyboard window handler establishing the location of the keyboard window on the touch sensitive panel and a set of message identifiers to allow the driver to communicate with the keyboard application. The driver uses the message identifiers to generate a message to the keyboard application in response to a mouse down event resulting from user contact within the keyboard window. A flag is also set by the driver so that subsequent mouse move events or a mouse up event result in the generation of messages or a message to the keyboard application.

[0013] It is also preferred that the keyboard application signals the computer to switch input focus to another application if input focus is given to the keyboard window.

[0014] According to another aspect of the present invention there is provided an interactive display system comprising:

an electronic writeboard;

a computer connected to said electronic writeboard, said computer including applications software and running an active application; and

a projector coupled to said computer and projecting the screen image of said computer onto said electronic writeboard wherein said electronic writeboard includes:

a touch sensitive panel on which the screen image output of said computer is displayed, said touch sensitive panel being responsive to user contact and generating events;

a keyboard window displayed on said touch sensitive panel and including a keyboard having a plurality of user selectable keys;

a driver receiving said events, said driver sensing user contact on said touch sensitive panel within said keyboard window and generating messages in response thereto; and a controller executing a keyboard application and receiving said messages, said keyboard application processing said messages to provide data to the active application running on said computer corresponding to keys of said keyboard contacted by said user.

[0015] According to still yet another aspect of the present invention there is provided in an electronic writeboard having a touch sensitive panel on which an on-screen keyboard is displayed, where user contact on said touch sensitive panel results in the generation of mouse events conveyed to a computer for processing, a method of inhibiting input focus being switched from an active application executed by said computer to said on-screen keyboard when said on-screen keyboard is touched, said method comprising the steps of:

detecting user contact on said touch sensitive panel;

forwarding events generated in response to contact on said touch sensitive panel outside of said on-screen keyboard directly to said computer; and

processing events generated in response to contact on said touch sensitive panel within said on-screen keyboard and forwarding said processed events to said active application.

[0016] The present invention provides advantages in that when the on-screen keyboard is used to enter data text into an active application, the active application does not flicker or close its open menus. This is achieved by inhibiting input focus from changing to the on-screen keyboard when it is touched. Also, since input focus does not change, text insertion points in active applications are not lost when the on-screen keyboard is used to enter data into the active application.

Brief Description Of The Drawings

[0017] An embodiment of the present invention will now be described more fully with reference to the accompanying drawings in which:

Figure 1 is a schematic view of an interactive display system including an electronic writeboard having a touch sensitive panel;

Figure 2 is a front view of the touch sensitive panel of Figure 1 on which an active application and an on-screen keyboard in accordance with the present invention are projected; and

Figure 3 is a flow chart illustrating the steps performed by the electronic writeboard when the on-screen keyboard is used to enter data into an active application.

Detailed Description Of The Preferred Embodiment

[0018] Referring now to Figure 1, an interactive display system is shown and is generally indicated to by reference numeral 10. As can be seen, interactive display system 10 includes an electronic writeboard 12 (EWB) of the type manufactured by SMART Technologies Inc. under model No. SB360 and sold under the name SMART Board. The SMART Board 12 includes a touch sensitive panel 14 and a tool tray 16 accommodating a plurality of tools 18 and having at least one user selectable button 19. The tools 18 include a number of colored pens and an eraser. A controller 28 having memory is installed in a slot of a personal computer 26 and executes a controller application to control the overall operation of the SMART Board 12. A driver in the form of an application is executed by the personal computer 26 and translates serial data from the controller 28 into events such as mouse events, tool change events and button press events.

[0019] The personal computer 26 is connected to a liquid crystal display panel 30 positioned on an overhead projector 32 so that the screen image presented on the monitor of the personal computer is projected onto the touch sensitive panel 14. As will be appreciated, the touch sensitive panel can be placed in front of a rear projection system.

[0020] The SMART Board 12 can operate either in a projection mode or in a 9 non-projection mode. As mentioned previously, in the projection mode, screen images generated by the personal computer 26 are projected onto the touch sensitive panel 14 and the SMART Board 12 functions as a giant mouse. In the non-projection modė, the touch sensitive panel 14 is mapped onto the drawing area of the running application so that writing and erasing on the touch sensitive panel 14 is stored as a computerized image and projected onto the touch sensitive panel.

[0021] When the electronic writeboard 12 and the personal computer 26 are connected and initialized, the electronic writeboard driver registers with the writeboard applications software executed by the personal computer. During this registration process, the driver determines the messages to which the applications software responds. Once the registration process has been completed, events generated by the SMART Board 12 as a result of user contact with the touch sensitive panel, selection of a tool, pressing of a button etc. are sent to the application software.

[0022] If the application software is not "aware" of the SMART Board 12 such as Microsoft NetMeeting i.e. the applications software does not use the SMART Board SDK source code, the personal computer 26 executes Aware interface software to interface the SMART Board 12 and the applications software. Specifics of the Aware interface software are described in pending U.S. Application No. 08/962,039 filed on October 31, 1997 and assigned to the assignee of the present invention, the contents of which are incorporated herein by reference.

[0023] In the preferred embodiment, the personal computer 26 runs a Windows® 95 platform and executes a variety of applications programs. In the projection mode, touching the touch sensitive panel results in mouse events being generated and processed by the personal computer. To allow a user to enter text data into an active application executed by the personal computer without having to use the keyboard of the personal computer 26, the controller 28 executes a keyboard application so that an on-screen keyboard 40 is presented within a window on the touch sensitive panel 14 (see Figure 2). As can be seen, the on-screen keyboard 40 includes a plurality of selectable keys 42 and in this embodiment is based on a "QWERTY" layout.

[0024] Unlike conventional prior art on-screen keyboards, the on-screen keyboard 40 in accordance with the present invention allows a user to enter data into an active application by touching the on-screen keyboard without input focus changing to the on-screen keyboard even though the personal computer 26 runs a Windows® platform. To achieve this, the driver and the controller 28 invoke a private communication mechanism so that mouse events generated as a result of contact with the touch sensitive panel 14 within the on-screen keyboard window are sent directly to the controller 28 instead of being routed through Windows®. In this manner, the code in Windows® responsible for the automatic activation and de-activation of applications in response to mouse events is bypassed. Thus, the controller 28 is able to process on-screen keyboard events without input focus changing. As a result, the problems associated with prior art on-screen keyboards described previously are overcome. Further specifics of the private communication mechanism established between the driver and the controller 28 will now be described with particular reference to Figure 3.

[0025] In operation, when the SMART Board 12 is initiated, the controller 28 registers the on-screen keyboard 40 with the driver by submitting two pieces of information to the driver. The first piece of information is the window handle of the keyboard window which includes its position on the touch sensitive panel 14. The second piece of information is a set of message identifiers and flags which allows the driver to communicate with the keyboard application executed by the controller 28.

[0026] Once the controller 28 has registered with the driver, when a user contacts the touch sensitive panel 14 and a mouse event is generated (block 48), the driver checks to see if the mouse down event occurred within the keyboard window (block 50). If the mouse down event occurred outside of the keyboard window, the driver simply sends the mouse event to Windows® (block 52). Windows® in turn handles the mouse down event in the conventional manner and the driver awaits the next mouse event.

[0027] If the mouse down event occurs within the keyboard window, the driver notifies the controller 28 of the mouse down event using the set of message identifiers and flags. The driver also sets a second flag so that all subsequent mouse events are sent to the keyboard application until a mouse up event is detected. When the controller 28 receives the identifiers and flags which represent the key selected by the user, the keyboard application proceeds to process the mouse down event and sends the selected key data to the active application with input focus. The text corresponding to the activated key 42 of on-screen keyboard is therefore, entered into the active application (blocks 54 and 56).

[0028] If a mouse move event or a mouse up event is generated, the driver checks to see if the flag was set to determine whether the event should be sent to the controller 28. If the flag is set, the mouse move or mouse up event is sent to the keyboard application for processing. If the flag is not set, the mouse event is sent to Windows®.

[0029] This process is repeated for each mouse down event so that mouse events occurring within the keyboard window bypass Windows® and thereby inhibit input focus changing from the active application to the on-screen keyboard 40. Source code for the driver and controller to perform the above-described operation is set out in Appendix A.

[0030] It is possible for the on-screen keyboard 40 to become active and gain input focus through use of an Alt-Tab hotkey sequence or through the Windows® Task Manager. If the on-screen keyboard 40 becomes active, the controller 28 automatically notifies Windows® to switch input focus to the next application in the Windows® task list. In this manner, if input focus is inadvertently switched to the on-screen keyboard 40, it is quickly switched to another application.

[0031] As will be appreciated, the present invention allows an on-screen keyboard to be used to enter text into an active application while avoiding the problems associated with prior art on-screen keyboards.

[0032] Although the electronic writeboard 12 has been described as a SMART Board including a tool tray with colored pens and an eraser, those of skill in the art will appreciate that the on-screen keyboard can be used with other electronic writeboards which use software tool panels allowing a user to select a tool. Also, although a preferred embodiment of the present invention has been described, those of skill in the art will appreciate that variations and modifications may be made without departing from the spirit and scope thereof as defined by the appended claims.

APPENDIX A

KEYBOARD APPLICATION

```
STRUCTURE registrationBlock
{
        WORD messageType;
        WORD downMessage;
        WORD upMessage;
        WORD moveMessage;
        WINDOW windowHandle;
};

WINDOW
smartBoardDriverWindow=FindWindow(driver_class_name,driver_title);
if (EXISTS(smartBoardDriverWindow))
{
        //create a block of data that we use to pass information to the driver
        registrationBlock myBlock;

        //The driver asks as what messages we would like to receive for the various
        //mouse movement events.
        myBlock.messageType=REGISTER_MOUSE;
        myBlock.downMessage=DOWN_MESSAGE;
        myBlock.upMessage=UP_MESSAGE;
        myBlock.moveMessage=MOVE_MESSAGE;
        myBlock.windowHandle=mainWindowHandle;

        //Use a special windows message called COPYDATA to send a block
        //of information to the driver
        SendMessage(smartBoardDriverWindow,WM_COPYDATA,&myBlock);
}

OnDownMessage(POSITION pos,FLAGS flags)
{
        savedbutton=GetButtonFromPosition(pos);
        if (savedbutton!=0)
        {
                savebutton.drawpusheddown();
        }
}

OnMoveMessage(POSITION pos,FLAGS flags)
{
        //No button was pressed, so ignore moves.
```

APPENDIX A (CON'T)

```
    if (savedbutton=0)
        return;

    BUTTON newbutton=GetButtonFromPosition(pos);
    if (newbuton!=savedbutton)
    {
        //move button up if mouse moves of it
        newbutton.Drawunpushedbutton();
        return;
    }
    else
    {
        newbutton.Drawpushedbutton();
    }
}

OnUpMessage(POSITION pos)
{
    //No button was pressed, so ignore the up
    if (savedbutton=0)
        return;
    if (GetButtonFromPosition(pos)!=savedButton)
        return;
    savedbutton.Drawunpushedbutton();
    savedbutton.SendKeyPressToWindows();
    savedbutton=0;
}
```

BOARD DRIVER APPLICATION

```
void HandleBoardRegisrationBlock(registrationBlock theBlock)
{
    //When we get the block from the keyboard asking for us to register it to get
    //mouse events directly
    //we store it in a list of these blocks so that many applications can use this
    //approach if needed.
    listOfRegisteredBlocks.AddToEnd(theBlock);
}

//This function is called when the board driver is about to send a mouse
//down to the system.
void HandleMouseDownEvent(POSITION pos,FLAGS flags)
```

APPENDIX A (CON'T)

```
{
    //What window are we pushing on.
    WINDOW PressedOn=WIndowFromPoint(pos);

    //is it one of our registered windows?
    for (INTEGER i=0;i<listOfRegisteredBlocks.GetNumberOfBlocks();i++)
    {
        registrationBlock
        block=listOfRegisteredBlocks.GetBlockByIndex(i);
        if (block.windowHandle=PressedOn)
        {
            //Press was on the registered window.  Send it the message

SendMessage(RegisteredWindow,block.downMessage,flags,pos);
            //save the handle so that all events until the up event can go to
            //the same place.
            blockForSubsequentEvents=block;
            return;
        }
    }
    //The window we are clicking on is not one of our registered windows, so
    //send the events directly to the operating system
    blockForSubsequentEvents=0;
    SendMessageToSystem(WM_LBUTTONDOWN,flags,pos);
}

void HandleMouseMoveEvent(POSITION pos,FLAGS flags)
{
    //If we sent the down event to a special window all subsequent events must go
    //there too
    if (blockForSubsequentEvents!=0)
    {
        //Press was on the registered window.  Send it the message
        SendMessage(blockForSubsequentEvents.windowHandle,
        blockForSubsequentEvents.downMessage,flags,pos);
    }
    else
    {
        SendMessageToSystem(WM_MOUSEMOVE,flags,pos);
    }
}

void HandleMouseUpEvent(POSITION pos,FLAGS flags)
```

APPENDIX A (CON'T)

```
{
        //If we sent the down event to a special window all subsequent events must go
        //there too
        if (blockForSubsequentEvents!=0)
        {
                //Press was on the registered window.  Send it the message
                SendMessage(blockForSubsequentEvents.windowHandle,

                blockForSubsequentEvents.upMessage,flags,pos);
        }
        else
        {
                SendMessageToSystem(WM_LBUTTONUP,flags,pos);
        }
        blockForSubsequentEvents=0;
                //reset it.  It will be initialized again on the next button down.
}
```